Section 2: Installing the DASD Component

Most of the DASD Component will be installed as normal part of installing CPExpert. However, up to three additional steps may be taken: (1) installing a modification to MXG or to NeuMICS to collect application-related or data set-related information, (2) defining workload categories, and (3) specifying data set information.

- **Collecting application-related or data set-related information**. This is an optional step. The DASD Component can optionally identify the specific applications (jobs and job steps) which access specific DASD volumes or use specific controllers or channel paths, or can analyze devices which provide a device I/O response time exceeding the response objective for selected data sets. This information can be quite useful in solving a DASD performance problem. With the application-related analysis, the DASD Component identifies the applications causing the problem or can identify the applications suffering from the problem. With the data set-related analysis, the DASD Component identifies devices which provide an I/O response time that exceeds a response objective for any critical data sets residing on the device. Providing this information is possible only if a modification is made to MXG or to MICS, to alter their processing of SMF Type 30(Data Definition) information.

  The modification included with CPExpert is quite simple.

  - For the application-related analysis, CPExpert records only the essential information required to identify the job name, the step name, the performance group number (for Compatibility Mode) or service class (for Goal Mode), the devices referenced and their SIO and connect times, and the beginning and end time of the measurement. This information is sufficiently concise that less than 25 cylinders of DASD are required to hold the information for all job steps executed in a relatively large installation. However, the information is sufficiently comprehensive for CPExpert to perform substantial analysis of DASD performance and the causes of poor DASD performance.

  - For the data set-related analysis, CPExpert records information associated with specific DD names: the job name, the step name, the performance group number or service class, the device referenced and the SIO and connect time for the device, and the beginning and end time of the measurement.

  Section 1 (Chapter 5) of this user manual describes the modifications and the processing of SMF information that CPExpert performs.

  Chapter 1 describes how to install the modification for MXG code, and Chapter 2 describes how to install the modification for MICS code. The installation for either product is very straightforward.

- **Defining workload categories**. This is an optional step. CPExpert allows you to define categories of work, and have the analysis focus on specific workload categories.

This can be useful, for example, if you wish to identify your "loved ones" and have the analysis treat this workload category as more important than other work.

Chapter 3 of this section describes how to define workload categories to CPExpert. |

- **Specifying data set names**. This is an optional step. CPExpert allows you to specify data set names and associate a response objective with each data set. CPExpert will analyze SMF data to detect periods when devices on which the data sets reside provide an I/O response greater than the response objective. If any such intervals are detected, CPExpert will perform a detailed analysis of the devices to identify why the device I/O response exceeded the response objective.

  Chapter 4 of this section describes how to define data set names and response objectives to CPExpert. |

The DASD Component provides many processing options (for example, you may exclude certain volumes from analysis, select particular time intervals for analysis, etc.). These processing options are not, strictly speaking, a part of the installation process. They may be employed as you refine your analysis of the performance of your DASD configuration. These processing options are described in Section 3 of this User Manual.

The instructions in this section assume that you have installed the CPExpert software. If you have not installed CPExpert, please install the software before continuing. (The procedures for installing CPExpert are described in the *CPExpert Installation Guide*.)

Chapter 1: Installing the modification for MXG

This chapter applies only if you use MXG to maintain your performance data base.

With normal MXG processing, the MXG TYPE30_D data set is not written because the data set would generally require **far** too much DASD space (often an entire volume would be required to hold the data set). Consequently, Barry Merrill does not write the TYPE30_D data set and he includes appropriate warnings in the IMAC30DD module that the data set may be very large. The CPExpert modification to MXG can allow you to collect essential information relating the DASD use to specific job steps and performance groups or service classes (with Goal Mode), without requiring massive amounts of DASD space. Additionally, the modification can allow you to collect information required to relate I/O activity to specific data sets.

Installing the modification for MXG is an optional step; however the DASD Component will be unable to relate DASD use and performance constraints to specific applications or to specific data sets unless you install the modification.

Implementing the DASD Component modification to MXG consists of five steps:

Step 1: Install the DASDMXG code.

Install the DASDMXG code into the SAS source library containing user modifications to MXG. This is done by copying the DASDMXG member from the CPEXPERT.SOURCE PDS into MXG.USERLIB (or whatever you call your library for user modifications to MXG). The DASDMXG member contains all the macros that will be invoked by the modifications described below.

Step 2: Modify MXG modules.

Modify one of two MXG modules to cause the DASD Component to output the DASD30DD data set. The MXG modules that may be modified are the IMACINTV module or the EXTY30U4 module.

- If you collect SMF Type 30 Interval records, modify the MXG module IMACINTV by including the following statement in the module:

```
%CPEOUTDD(V);      /* CPEXPERT MODIFICATION */
```


Exhibit 2-1 illustrates the MXG IMACINTV module with the modification.

```
/*  COPYRIGHT (C) 1985,1992 BY MERRILL CONSULTANTS DALLAS TEXAS     */
/*                                                                  */
/* IMACINTV CONTROLS THE CREATION OF TYPE30_V  DATASET, BUILT FROM  */
/* TYPE 30 INTERVAL RECORDS (SUBTYPES 2 AND 3).                     */
/*  NOTE THAT THE PDB.SMFINTRV DATASET IS A RENAME OF TYPE30_V AND  */
/*  THUS CHANGING THIS MEMBER TO CREATE TYPE30_V OBSERVATIONS WILL  */
/*  ALSO CREATE PDB.SMFINTRV IF YOU EXECUTE BUILDPDB/BUILDPD3.      */
/* BY DEFAULT, THIS MEMBER CREATES NO OBSERVATIONS IN TYPE30_V.     */
/* IF YOU WANT ANY OBSERVATIONS TO BE CREATED IN THE DATASET, YOU   */
/* MUST EXECUTE AN   OUTPUT _LTY30UV;  STATEMENT IN THIS MEMBER.    */
/* FOR  EXAMPLE, IF YOU WANT AN OVSERVATION IN TYPE30_V FOR EACH AND*/
/* EVERY TYPE 30 SUBTYPE 2 OR SUBTYPE 3 RECORD READ, SIMPLY REMOVE  */
/* THE COMMENT BLOCK AROUND THE FOLLOWING STATEMENT:                */
/* FOLLOWING STATEMENT:                                             */
/*                                                                  */
/*         OUTPUT _LTY30UV ;                                        */
/*                                                                  */
/* THE _LTY30UV MACRO, DEFINED IN IMAC30, DEFAULTS TO TYPE30_V.     */
/*                                                                  */
/* ALTERNATIVELY, IF YOU  WISH  TO  CREATE  OBSERVATIONS  ONLY  FOR */
/* STARTED  TASKS,  REMOVE  THE  COMMENT  BLOCK  FROM THE FOLLOWING */
/* STATEMENT:                                                       */
/*                                                                  */
/*         IF TYPETASK='STC' THEN OUTPUT _LTY30UV;                  */
/*                                                                  */
/* IMACINTV IS %INCLUDED AFTER THE "IMPORTANT" VARIABLES, SUCH AS   */
/* JOB, READTIME, STEPNAME, PROGRAM, TYPETASK, JOBCLASS, SMFTIME,   */
/* ETC. HAVE BEEN DEFINED.  THUS YOUR SELECTION CRITERIA CAN CAUSE  */
/* TYPE30_V TO CONTAIN OBSERVATIONS ONLY  FROM SPECIFIC JOBS AND/OR */
/* PARTICULAR PROGRAM NAMES, SUCH AS                                */
/*                                                                  */
/*    IF JOB=:'CICS' OR PROGRAM='DFHSIP' THEN OUTPUT _LTY30UV;      */
/*                                                                  */
/* REALIZE THAT THIS MEMBER CONTROLS ONLY THE SELECTION OF TYPE 30  */
/* INTERVAL RECORDS. YOU MUST (IN PARMLIB) SPECIFY THAT YOU DESIRE  */
/* INTERVAL RECORDS TO BE CREATED BY SMF; YOU CAN SPECIFY SEPARATE  */
/* DURATIONS INDIVIDUALLY FOR JOBS (JOB), TSO SESSIONS (TSU), AND   */
/* OR STARTED TASKS (STC).                                          */
/*                                                                  */
/*                                                                  */
/*                                                                  */
/*                                                                  */
/********************************************************************/
/***   THIS MODULE IS INCLUDED BY THE FOLLOWING MXG MODULES:      ***/
/***                                                              ***/
/***     VMAC30                                                   ***/
/***                                                              ***/
/********************************************************************/

%CPEOUTDD(V);           /* CPEXPERT MODIFICATION      */
```

MXG IMACINTV MODULE, WITH CPEXPERT MODIFICATION

EXHIBIT 2-1

MXG coding printed with permission of Merrill Consultants, Dallas, Texas.

The macro parameter "(V)" in the above code causes SAS statements to be generated to identify the source of the data (namely, the SMF Type 30 Interval records).

Collecting interval data is BY FAR the preferred approach, since much more comprehensive analysis can be accomplished.

- If you do **not** collect interval data but collect only Workload Termination data (that is, you collect SMF Type 30 Workload Termination records but do not collect Type 30 Interval records), you can still obtain information about the DASD use of specific job steps. However the analysis will not be as comprehensive as if you collect interval information.

  If you collect only Workload Termination records, modify the MXG module EXTY30U4 by including the following statement in the module:

```
%CPEOUTDD(4);    /* CPEXPERT MODIFICATION    */
```

The macro parameter "(4)" in the above code causes SAS statements to be generated to identify the source of the data (namely, the SMF Type 30 Workload Termination records.

Exhibit 2-2 illustrates the MXG EXTY30U4 module with the modification.

```
/* COPYRIGHT (C) 1985,1992 BY MERRILL CONSULTANTS DALLAS TEXAS USA */
/*****************************************************************/
/*  MEMBER  EXTY30U4  -  MXG  OUTPUT EXIT FOR DATA SET TYPE30_4  */
/*****************************************************************/
/*  THIS MEMBER CONTAINS THE OUTPUT STATEMENT FOR THE DATA SET.  */
/*  CHANGES IN DDNAME, DSNAME, AND THE CREATION OF NEW VARIABLES */
/*   CAN BE CODED HEREIN.                                        */
/*****************************************************************/
  OUTPUT _LTY30U4; /* TYPE30_4, DEFINED IN IMAC30 */

 %CPEOUTDD(4);                  /* CPEXPERT MODIFICATION       */
```

MXG EXTY30U4 MODULE, WITH CPEXPERT MODIFICATION

EXHIBIT 2-2

MXG coding printed with permission of Merrill Consultants, Dallas, Texas.

Step 3: Modify MXG IMAC30DD module.

Modify the MXG IMAC30DD module to cause the DASD Component to collect the Data Definition information that is contained in the DASD30DD data set. This is accomplished by including the following statement in the module:

```
%DASD_MXG;    /* CPEXPERT MODIFICATION    */
```

Exhibit 2-3 illustrates the MXG IMAC30DD module with the modification.


```
/* COPYRIGHT (C) 1985,1992 BY MERRILL CONSULTANTS DALLAS TEXAS */
/**********MEMBER=IMAC30DD************************************/
/*                                                            */
/* THIS MODULE CONTROLS THE EXISTENCE OF DATA SET   TYPE30_D  */
/*  (WHICH IS RENAMED PDB.DDSTATS BY BUILDPDB/BUILDPDE).      */
/*                                                            */
/* BECAUSE TYPE30_D COULD CONTAIN ONE OBSERVATION FOR EVERY DD */
/*  CARD IN EVERY SUBTYPE OF EVERY TYPE 30 RECORD, THE DATASET */
/*  COULD BE VERY, VERY LARGE, AND THUS BY DEFAULT, MXG DOES NOT */
/*  CREATE ANY OBSERVATIONS IN TYPE30_D.                      */
/*                                                            */
/* YOU MUST MODIFY THIS INSTALLATION MACRO TO CREATE OBSERVATIONS! */
/*                                                            */
/* NOTE THAT DATASET TYPE30_D AND PDB.DDSTATS WILL ALWAYS EXIST; */
/*  THIS EXIT CONTROLS ONLY IF THERE ARE OBSERVATIONS CREATED. */
/*                                                            */
/* THE DSNAME MACRO _LTY30UD IS DEFINED IN IMAC30 AND DEFAULTS */
/*  TO TYPE30_D.                                              */
/*                                                            */
/* THE FOLLOWING EXAMPLE HAS BEEN COMMENTED OUT SO THAT IT IS NOT */
/*  EXECUTED, AND IT SHOWS ONE OF MAY CONDITIONS YOU MIGHT CHOOSE */
/*  TO RESTRICT WHICH DD SEGMENT'S OBSERVATIONS ARE OUTPUT.  THE */
/*  EXAMPLE WILL CREATE OBSERVATIONS ONLY FOR THE STEP TERMINATION */
/*  (SUBTYPE=4) RECORD, AND ONLY FOR THE SPECIFIC DEVICE NUMBER */
/*  (UCB ADDRESS=38F) IN THIS EXAMPLE.  YOU COULD CHOSE TO TEST */
/*  FOR JOB, STEP, DDNAME, OR ANY OF THE VARIABLES IN TYPE30. */
/*                                                            */
/* NOTE: THE EXAMPLE TEST IS INSIDE A PAIR OF COMMMENT SYMBOLS */
/*       WHICH PRECEED AND FOLLOW THE EXAMPLE "IF STATEMENT". */
/*       YOU MUST REMOVE THE COMMENT SYMBOLS AROUND THE TEST, AND */
/*       CHANGE THE TEST AS DESIRED, TO CREATE OBSERVATIONS.  */
     /*
       IF DEVNR=038FX AND SUBTYPE=4 THEN OUTPUT _LTY30UD;
     */

/*************************************************************/
/***   THIS MODULE IS INCLUDED BY THE FOLLOWING MXG MODULES:  ***/
/***                                                          ***/
/***     VMAC30                                               ***/
/***                                                          ***/
/*************************************************************/
/*                                                            */

%DASD_MXG;  * CPEXPERT MODIFICATION TO ACQUIRE DASD INFORMATION;
```

MXG IMAC30DD MODULE, WITH CPEXPERT MODIFICATION

EXHIBIT 2-3

MXG coding printed with permission of Merrill Consultants, Dallas, Texas.

Step 4(alternate): Modify MXG EXPDBINC module and EXPDBVAR module.

This step applies if you use the standard MXG BUILDPDB or use SAS/ITSV to process SMF data.

Modify the MXG EXPDBINC module as shown below:

```
%INCLUDE SOURCLIB(DASDMXG); /* CPExpert modification */
```

Modify the MXG EXPDBVAR module as shown below:

```
MACRO _VARUSER      /* CPExpert modification */
   %%VARDDCPE(MAXDASD=nn,MAXDD=nnn,MINIO=nnn,COLLECT=xxxxxxx)
%
```

The %%VARDDCPE macro statement generates code to define the DASD30DD data set, define the DASD30DS data set, and describe the variables contained in the KEEP statement for the data sets. (**NOTE**: two "%" symbols are required since the code is placed into SAS "old style" macros.) The macro parameters specify the maximum number of unique DASD devices referenced by any job step in your environment, the maximum number of DD statements associated with critical data sets, the significant number of I/O operations directed to any data set in any Type 30(Interval) record, and which type of data collection should be performed.

- The **MAXDASD** macro parameter specifies the maximum number of unique DASD devices that are referenced by any job step in your environment. Note that this is **not** the number of DD statements. Rather, this is the number of unique devices. There often will be many DD statements for a single device, but the DASD_MXG software combines these DD statements into one observation in the DASD30DD data set.

  The default specification for the MAXDASD macro parameter is **MAXDASD=25**. The DASD_MXG software will produce a warning message if this value is too low, and the software simply ignores devices for any job step exceeding the value specified. (There is no other effect; that is, the software will not abort and the analysis software will continue to function properly.)

  The only reason for specifying 25 versus 200 (for example) as an initial value, is that the larger the value, the more DASD space that will be used for the DASD30DD data set. The record size will increase by 15 bytes for each unique device specified. If you process 20,000 job steps each day and specify a maximum of 25 unique devices per job step, the total temporary DASD space required for the DASD30DD data set will be about 25 cylinders (3380). This is not a large amount in any event, but there is no point in writing larger records than are needed.

Therefore, MAXDASD=25 should be specified initially and the value should be increased only if required.

- The **MAXDD** macro parameter specifies the maximum number of unique DD names that may be associated with critical data sets. As described in Section 1 (Chapter 5.3), the CPExpert modification to MXG creates an array which contains the DD names of all DD statements associated with data sets which have been defined to CPExpert with a response objective. (The array is created only if you specify DDNAMES or BOTH in the COLLECT macro parameter described below.)

  Each array element requires only 8 bytes of storage. This amount storage is insignificant. However, SAS generates a SAS data element name for each array element. SAS 5.18 has a restriction on the total number of data element names which can be defined. When this number is exceeded, SAS aborts with an error in the compilation stage. Therefore, you cannot set an extremely large value for the MAXDD parameter when executing under SAS 5.18.

  The default specification for the MAXDD macro parameter is **MAXDD=200**, generating an array capable of containing 200 unique DD names. The DASD_MXG software will produce a warning message if the number of unique DD names is greater than the array size. The software will list all DD names which it could not process, and it simply ignores DD names for any job step exceeding the value specified. (There is no other effect; that is, the software will not abort and the analysis software will continue to function properly.)

  If you receive a warning message from the CPExpert modification, you should consider (1) increasing the value of the MAXDD macro parameter or (2) restricting the number of data sets you define as critical.

- The **MINIO** macro parameter specifies the minimum number of I/O operations to any data set within a particular SMF Type 30(Interval) record. The CPExpert modification will ignore any data sets encountered if the number of I/O operations is below the MINIO value. The point of this parameter is that you probably will not worry about data sets with a very low activity, regardless of the DASD response time they experienced. There is no point in collecting information about very low activity data sets.

  The default specification for the MINIO macro parameter is **MINIO=100**.

- The **COLLECT** macro parameter specifies whether to collect device information (required to associate device activity with application systems), to collect DD name information (required to associate device activity with data sets), or to collect both device and DD name information.

Specify **COLLECT=DEVICES** to collect device information. This is the default setting. If you have previously installed the modification for MXG, you do not have to make any changes to your installation.

Specify **COLLECT=DDNAMES** to collect DD name information.

Specify **COLLECT=BOTH** to collect both device information and DD name information.

WARNING: Please review Chapter 6 of this section before specifying DDNAMES or BOTH as an option for the COLLECT parameter. You should **not** specify either of these options unless you have defined critical data sets and have followed the procedures in Chapter 6.

Step 4(alternate): Modify the SAS job stream used to execute MXG.

This step applies if you use in-stream JCL for MXG to process SMF data. Modify the SAS job stream used to execute MXG by making the following simple modifications:

- Modify the %INCLUDE statement to add **DASDMXG** to the list of included code.
- Add the following macro statement after the MXG DATA statement:

%VARDDCPE(MAXDASD=nn,MAXDD=nnn,MINIO=nnn,COLLECT=xxxxxxx)

Please refer to the discussion in "Step 4 (if you use the standard MXG BUILDPDB to process SMF data)" for a description of the macro variables.

WARNING: Do not include a semicolon after the macro invocation!

Exhibit 2-4 illustrates a sample MXG SAS job stream with the three modifications highlighted. Of course, your own MXG SAS job stream may appear somewhat different than that shown in Exhibit 2-4. However, your MXG SAS job stream should be sufficiently similar to Exhibit 2-4 that you can appreciate how the modifications should appear in your own MXG SAS job stream.

Step 5: Add CPEDASD DD statement to the JCL.

Add the CPEDASD DD statement to the job control language (JCL) used to execute MXG. The CPEDASD DD statement defines the SAS library in which the CPExpert modification will place the DASD30DD data set or DASD30DS data set created by the modification.

The CPEDASD DD statement can refer to the CPEDATA SAS library if you wish, or you can allocate a different SAS library. For space allocation purposes, the CPEDASD data set requires slightly less than one cylinder of DASD space per 2,200 SMF Type 30(Interval) records processed by MXG. For example, if you typically have about 22,000 Type 30(Interval) records in your daily SMF file, you will need about 10 cylinders of DASD allocated to the CPEDASD SAS library. It is difficult to estimate the amount of space required for the DASD30DS data set, since this is a function of the number of critical data sets you define and the number of unique DD names associated with the data sets. This data set contains about 730 observations per track.

Exhibit 2-4 illustrates a sample MXG SAS job stream with the CPEDASD DD statement included, using the CPEDATA SAS library space.

```
//CPEDASD     DD      DSN=prefix.CPEXPERT.CPEDATA,DISP=OLD
//SYSIN       DD      *

%INCLUDE
 SOURCLIB(VMACSMF,VMAC7072,VMAC71,VMAC73,VMAC74,VMAC75,VMAC77,
          VMAC78,VMAC30,VMACTSOM,VMAC434,VMAC40 VMAC434D,
          DASDMXG);
DATA

  %VARDDCPE(MAXDASD=25,MAXDD=200,MINIO=100,COLLECT=DEVICES)

 _VAR7072
 _VAR71


etc.


;
```

SAMPLE MXG SAS JOB STREAM

EXHIBIT 2-4

Chapter 2: Installing the modification for NeuMICS

This chapter applies only if you use MICS to maintain your performance data base.

The normal MICS processing summarizes the device information from the SMF Type 30(DD) EXCP segment into the BATWDA dataset. The BATWDA dataset is created only if specified during the MICS installation process. The BATWDA data set often is not created because it is so large, and the information is often too general for detailed performance analysis. CPExpert does not use the MICS BATWDA file. Rather, CPExpert uses data created by a proprietary modification to MICS. The CPExpert modification to MICS is described in this section.

The CPExpert modification to MICS allows you to collect essential information relating the DASD use to specific job steps and performance groups or service classes (with Goal Mode), without requiring massive amounts of DASD space. Additionally, the modification allows you to collect information required to relate I/O activity to specific data sets. The SAS data sets created by the modification to MICS normally are small (typically about 10 cylinders of DASD), and they are easy to process.

The CPExpert modification to MICS invokes two standard MICS user exits and includes a simple (one-line) modification to the DYSMFFMT module. Installing the modification for MICS is an optional step; however the DASD Component will be unable to relate DASD use and performance constraints to specific applications or to specific data sets unless you install the modification.

Implementing the CPExpert modification to MICS consists of five steps:

Step 1: Install the DASDMIC code.

Install the DASDMIC code into the prefix.MICS.USER.SOURCE library containing user modifications to MICS. This is done by copying the DASDMIC member from the CPEXPERT.SOURCE partitioned data set into the prefix.MICS.USER.SOURCE library. The DASDMIC member contains all the macros that will be invoked by the modifications described below.

Step 2: Modify the sharedprefix.MICS.USER.SOURCE(#SMFEXIT).

Modify the sharedprefix.MICS.USER.SOURCE(#SMFEXIT), by including SAS statements (1) to include the CPExpert macros, (2) to define the _USRDMAP macro, and (3) to define the _USRSSF macro.

Exhibit 2-5 illustrates the normal sharedprefix.MICS.USER.SOURCE(#SMFEXIT) member before the modifications. Exhibit 2-6 illustrates the normal sharedprefix.MICS.USER.SOURCE(#SMFEXIT) member after the modifications.

```
/* some comments */
%INCLUDE SOURCE(#SMFEXIT);
```

NORMAL sharedprefix.MICS.USER.SOURCE(#SMFEXIT) BEFORE MODIFICATION

EXHIBIT 2-5

```
/* some comments */
%INCLUDE SOURCE(#SMFEXIT);

/* CPEXPERT MODIFICATION */

%INCLUDE USOURCE(DASDMIC);

MACRO _USRDMAP
/* DEVICE ADDRESS MAPPING EXIT */
%%DASD_MIC;
%

 MACRO _USRSSFS
/*  STEP TERMINATION INTERIM SMF FILE EXIT */
%%CPEOUTDD;
%
```

NORMAL sharedprefix.MICS.USER.SOURCE(#SMFEXIT) AFTER MODIFICATION

EXHIBIT 2-6

WARNING: Please note that two (2) "%" characters are used when invoking "new style" macros inside "old style" macros in the user exits.

- The **%INCLUDE USOURCE(DASDMIC);** statement shown in Exhibit 2-6 causes MICS to include the DASDMIC macros as MICS loads its own code.

- The **%%DASD_MIC;** SAS macro statement shown as part of the _USRDMAP macro in Exhibit 2-6 generates code and arrays to process and store information related to job step use of DASD, by device address.

- The **%%CPEOUTDD;** SAS macro statement shown as part of the _USRSSFS macro in Exhibit 2-6 generates code to output the CPEDASD.DASD30DD data set, and clears the arrays described above.

The effect of Step 2 is that the normal _USRDMAP and _USRSSFS user exits included as null members in sharedprefix.MICS.SOURCE(#SMFEXIT) will be overridden by the code required for CPExpert.

Step 3: Modify prefix.MICS.SOURCE(DYSMFFMT).

Modify the sharedprefix.MICS.SOURCE(DYSMFFM1) module[1] by including the following SAS statement just after the "DATA" statement:

```
%VARDDCPE(MAXDASD=nn,MAXDD=nnn,MINIO=nnn,COLLECT=xxxxxxx)
```

WARNING: Do not include a semicolon after the macro invocation!

The %VARDDCPE macro statement generates code to define the DASD30DD data set, define the DASD30DS data set, and describe the variables contained in the KEEP statement for the data sets. The macro parameters specify the maximum number of unique DASD devices referenced by any job step in your environment, the maximum number of DD statements associated with critical data sets, the significant number of I/O operations directed to any data set in any Type 30(Interval) record, and which type of data collection should be performed.

- The **MAXDASD** macro parameter specifies the maximum number of unique DASD devices that are referenced by any job step in your environment. Note

[1] It is unfortunate that a modification must be made to the MICS coding, since this is contrary to standards implemented at many installations. A request was made to LEGENT Corporation (through the MICS Advisory Council, as Incident #441496) that a standard user exit be implemented after the DATA statement. If this request had been accepted and implemented, output files could be generated easily within the standard MICS user exit facility. Using a standard MICS user exit facility would obviate the need for a modification to MICS code. Computer Associates has not indicated a desire to make the modification.

that this is **not** the number of DD statements. Rather, this is the number of unique devices.

There often will be many DD statements for a single device, but the DASD_MIC software combines these DD statements into one observation in the DASD30DD data set.

The default specification for the MAXDASD macro parameter is **MAXDASD=25**. The DASD_MIC software will produce a warning message if this value is too low, and the software simply ignores devices for any job step exceeding the value specified. (There is no other effect; that is, the software will not abort and the analysis software will continue to function properly.)

The only reason for specifying 25 versus 200 (for example) as an initial value, is that the larger the value, the more DASD space that will be used for the DASD30DD data set. The record size will increase by 15 bytes for each unique device specified. If you process 20,000 job steps each day and specify a maximum of 25 unique devices per job step, the total temporary DASD space required for the DASD30DD data set will be about 25 cylinders (3380). This is not a large amount in any event, but there is no point in writing larger records than are needed. Therefore, MAXDASD=25 should be specified initially and this value should be increased only if required.

- The **MAXDD** macro parameter specifies the maximum number of unique DD names that may be associated with critical data sets. As described in Section 1 (Chapter 5.3), the CPExpert modification to MICS creates an array which contains the DD names of all DD statements associated with data sets which have been defined to CPExpert with a response objective. (The array is created only if you specify DDNAMES or BOTH in the COLLECT macro parameter described below.)

  Each array element requires only 8 bytes of storage. This amount storage is insignificant. However, SAS generates a SAS data element name for each array element. SAS 5.18 has a restriction on the total number of data element names which can be defined. When this number is exceeded, SAS aborts with an error in the compilation stage. Therefore, you cannot set an extremely large value for the MAXDD parameter when executing under SAS 5.18.

  The default specification for the MAXDD macro parameter is **MAXDD=200**, generating an array capable of containing 200 unique DD names. The DASD_MIC software will produce a warning message if the number of unique DD names is greater than the array size. The software will list all DD names which it could not process, and it simply ignores DD names for any job step exceeding the value specified. (There is no other effect; that is, the software will not abort and the analysis software will continue to function properly.) If you

receive a warning message from the CPExpert modification, you should consider (1) increasing the value of the MAXDD macro parameter or (2) restricting the number of data sets you define as critical.

- The **MINIO** macro parameter specifies the minimum number of I/O operations to any data set within a particular SMF Type 30(Interval) record. The CPExpert modification will ignore any data sets encountered if the number of I/O operations is below the MINIO value. The point of this parameter is that you probably will not worry about data sets with a very low activity, regardless of the DASD response time they experienced. There is no point in collecting information about very low activity data sets.
  The default specification for the MINIO macro parameter is **MINIO=100**.

- The **COLLECT** macro parameter specifies whether to collect device information (required to associate device activity with application systems), to collect DD name information (required to associate device activity with data sets), or to collect both device and DD name information.

  Specify **COLLECT=DEVICES** to collect device information. This is the default setting. If you have previously installed the modification for MICS, you do not have to make any changes to your installation.

  Specify **COLLECT=DDNAMES** to collect DD name information.

  Specify **COLLECT=BOTH** to collect both device information and DD name information.

  WARNING: Please review Chapter 6 of this section before specifying DDNAMES or BOTH as an option for the COLLECT parameter. You should **not** specify either of these options unless you have defined critical data sets and have followed the procedures in Chapter 6.

Exhibit 2-7 illustrates the prefix.MICS.SOURCE(DYSMFFM1) code with the modification, reflecting explicit stating of the default values.


```
DATA

 %VARDDCPE(MAXDASD=25,MAXDD=200,MINIO=100,COLLECT=DEVICES)

 %MAINWRK(CCC=SMF,WRK=INI)
```

SAMPLE sharedprefix.MICS.SOURCE(DYSMFFM1)
WITH CPEXPERT MODIFICATION

EXHIBIT 2-7

Step 4: Add the CPEDASD DD statement to the JCL.

Add the CPEDASD DD statement to the job control language (JCL) used to execute MICS. The CPEDASD DD statement defines the SAS library in which the CPExpert modification will place the DASD30DD and DASD30DS data sets created by the modification.

Exhibit 2-8 illustrates a sample CPEDASD DD statement.

```
//CPEDASD DD DSN=prefix.CPEXPERT.CPEDATA,DISP=OLD
```

SAMPLE CPEDASD DD STATEMENT IN MICS JCL

EXHIBIT 2-8

The CPEDASD DD statement can refer to the CPEDATA SAS library if you wish, or you can allocate a different SAS library. For space allocation purposes the CPEDASD data set requires slightly less than one cylinder of DASD space per 2,200 SMF Type 30(Interval) records processed by MXG or MICS to contain the application-related information written to the DASD30DD SAS data set (that is, if you have specified COLLECT=DEVICES or COLLECT=BOTH). For example, if you typically have about 22,000 Type 30(Interval) records in your daily SMF file, you will need about 10 cylinders of DASD allocated to the CPEDASD SAS library to contain the application-related information.

It is difficult to estimate the amount of space required for the data set-related information written to the DASD30DS data set, since this is a function of the number of critical data sets you define and the number of unique DD names associated with the data sets. The DASD30DS data set contains about 730 observations per track.

MICS users may wish to place the CPEDASD DD statement into PROCLIB and then do a MICS JCLGEN to copy the information into the MICS prefix.CNTL libname.

We STRONGLY suggest that you test the modification in a MICS Test environment before placing the modification into a MICS production environment! If you (or we) have made a mistake, the MICS production run might abort. This certainly would not generate a favorable impression about CPExpert.

Chapter 3: Defining workload categories

The Workload Definition Section of the DASGUIDE module can be used to (1) define workload categories, (2) associate performance groups with the workload categories prior to MVS (Goal Mode), (3) associate service classes with the workload category with MVS (Goal Mode), (4) assign a relative importance to the workloads, and (5) direct CPExpert to perform analysis based upon the defined workload categories.

The workload definition is optional; you do not have define any workloads. However, you must define workloads if you wish CPExpert to analyze the DASD performance effects of contending workloads. Exhibit 2-9 illustrates a sample workload definition.

```
*********************************************************************** ;
*  WORKLOAD DEFINITION BY PERFORMANCE GROUP                             ;
*********************************************************************** ;
* DO NOT REMOVE THE FOLLOWING SAS MACRO COMMENT LINE!                   ;
/* WORKLOAD DEFINITION
BEGIN THE WORKLOAD DEFINITION;
BATCH= 1             * SAMPLE: BATCH IS PGN 1                         ;
TSO  = 2,6,9         * SAMPLE: TSO IS PGN 2,6,9                       ;
CICS = 12,14         * SAMPLE:  CICS IS PGN 12,14                     ;
* DO NOT REMOVE THE FOLLOWING MACRO COMMENT LINE!                      ;
*/
*********************************************************************** ;

*********************************************************************** ;
*  WORKLOAD DEFINITION BY SERVICE CLASS
      ;
*********************************************************************** ;
* DO NOT REMOVE THE FOLLOWING SAS MACRO COMMENT LINE!                   ;
/* WORKLOAD DEFINITION (GOAL MODE) BEGIN THE WORKLOAD DEFINITION;
BATCH = BATHI,BATMED,BATLOW
CICS        = CICSAOR1,CICSAOR2,CICSAOR3
* DO NOT REMOVE THE FOLLOWING MACRO COMMENT LINE!                      ;
*/
```

SAMPLE DISPLAY OF CPEXPERT.USOURCE(DASGUIDE) MODULE

EXHIBIT 2-9

he workload category definition syntax **prior** to MVS (Goal Mode) is:

NAME = PGN1[,PGN2...,PGNn]

- **NAME** is the workload name (e.g., BATCH, TSO, CICS, etc.). The name an be any number of characters (including blanks), since it is delimited by the equal sign. However, you should generally restrict the length to 10 characters, since some reports produced by CPExpert assume a maximum length for the column containing the workload name. No more than 10 workload category names may be defined. A workload name must not begin with "*".

- **PGN** is the number of the performance group(s) associated with this workload category. The performance group numbers are separated by commas, and the entire set may optionally be enclosed in parentheses. Up to 10 performance groups may be associated with a workload category[2]. Any performance groups not associated with a workload category are placed in the OTHER category.

The workload category definition syntax **with** MVS (Goal Mode) is:

NAME = SRV1[,SRV2...,SRVn]

- **NAME** is the workload name (e.g., BATCH, TSO, CICS, etc.). The name an be any number of characters (including blanks), since it is delimited by the equal sign. However, you should generally restrict the length to 10 characters, since some reports produced by CPExpert assume a maximum length for the column containing the workload name. No more than 10 workload category names may be defined. A workload name must not begin with "*".

- **SRV** is the service class name(s) associated with this workload category. The service class names are separated by commas, and the entire set may optionally be enclosed in parentheses. Up to 10 service classes may be associated with a workload category[3]. Any service classes not associated with a workload category are placed in the OTHER category.

NOTE: If you are running Goal Mode, please be sure that you have modified USOURCE(GENGUIDE) to specify **%LET GOALMODE=Y;** so the CPExpert code related to service classes will be generated.

[2]The limits of 10 different workload categories and 10 performance groups per category were selected because they seem so large that they should satisfy your performance analysis needs. Please give us a call if you have a requirement for more than these limits.

[3]The limits of 10 different workload categories and 10 service classes per category were selected because they seem so large that they should satisfy your performance analysis needs. Please give us a call if you have a requirement for more than these limits.

Summary of Coding Restrictions:

- A workload name must **not** begin with "*".
- A maximum of 10 workload categories may be defined.
- A maximum of 10 performance groups or service classes may be assigned to any workload category.
- The entire definition for a workload category must be contained within a single line[4].
- You may define the same performance groups or service classes in more than one workload category. However, the analysis may give strange results if the workload categories are defined as "competing" with each other (you have the strange situation of a performance group or service class competing with itself!).

There is a potentially significant limitation with defining workload categories: workload categories can be identified only by performance group or service class rather than performance group or service class **period**. This limitation is imposed by the SMF data; the standard SMF data associates performance or resource use to performance group periods or service class periods only in the SMF Type 72 records. It is not possible to associate DASD use, for example, to specific performance group or service class periods.

[4]This did not seem to be a particularly restrictive design constraint, so it didn't seem necessary to incorporate "continuation line" logic. If this does pose a problem, please give Computer Management Sciences a call.

Chapter 4: Defining critical data sets

The Data Set Definition Section of the DASGUIDE module can be used to (1) define data sets and (2) associate a response objective with each data set.

The data set definition is optional; you do not have define any data sets. However, you must define data sets and specify a response objective for each data set if you wish CPExpert to analyze the DASD performance provided to critical data sets.

Exhibit 2-10 illustrates a sample data set definition.

```
*****************************************************************;
*  DATA SET NAME/RESPONSE OBJECTIVE DEFINITIONS
          ;
*****************************************************************;
* DO NOT REMOVE THE FOLLOWING SAS MACRO COMMENT LINE!           ;
/* DEFINE DATA SET NAMES                                        ;
DSN=USER.EXTERNAL.DATA,RESPONSE=20                              ;
DSN=USER.INTERNAL,RESPONSE=30   ;
DSN=USER.INTERNAL.EXECUTIVE.DATA,RESPONSE=5;
DSN=USER.INTERNAL.SYSTEMS.DATA,RESPONSE=20;
* DO NOT REMOVE THE FOLLOWING MACRO COMMENT LINE!          ;
*/
*****************************************************************;
```

SAMPLE DISPLAY OF CPEXPERT.USOURCE(DASGUIDE) MODULE

EXHIBIT 2-10

The data set definition syntax is:

DSN = data.set.name,RESPONSE=obj

- **DSN** is the data set name. The data set name can be any valid data set name. The processing of data set name "wild cards" and the order in which data set names are processed are described below.

- **RESPONSE** is the device response objective for the identified data set. This specification is in milliseconds. For example, RESPONSE=20 means that the

device on which the data set resides should provide an average I/O response of 20 milliseconds during any RMF measurement interval. A response objective must be specified for each data set defined.

CPExpert places each data set name into SAS macro variables, and places the associated response objective into macro variables corresponding to the data set name.

- CPExpert processes the data set names with an implicit trailing "wild card" specification. That is, if only high level qualifiers are specified, all data set names beginning with the high level qualifiers are considered to be associated with the specified response objective.

- The order of appearance in the list controls the application of response objectives.

Examine Exhibit 2-10 to appreciate the significance of the above rules.

- All data sets beginning with "USER.EXTERNAL.DATA" will have a response objective of 20 milliseconds.

- All data sets beginning with "USER.INTERNAL" will have a response objective of 30 milliseconds. However, any data set beginning with "USER.INTERNAL.EXECUTIVE.DATA" will have a response objective of 5 milliseconds, while any data set beginning with "USER.INTERNAL.SYSTEMS.DATA" will have a response objective of 20 milliseconds.

- If the statement "DSN=USER.INTERNAL,RESPONSE=30" in Exhibit 2-10 were to be placed at the end of the list, it would override the previous statements and all data sets beginning with "USER.INTERNAL" would have a response objective of 30 milliseconds, regardless of whether they were associated with "EXECUTIVE.DATA" or with "SYSTEMS.DATA".

WARNING: Do not specify response objectives for data sets unless they are critical to accomplishing your management objectives, and do not make excessive use of the "wild card" technique. The purpose of this warning is to alert you to the fact that significant processing time and DASD space could be required if you do not judiciously apply this analysis technique.

The following operational notes apply to defining critical data sets:

- The DAS1415 module of CPExpert must be executed before you execute MXG or MICS to perform your normal daily update of your performance data base. The DAS1415 module processes the data set name and response objectives you have defined in USOURCE(DASGUIDE). The DAS1415 module then processes the raw SMF Type 14/15 records to identify jobs which reference the data sets, and to build a SAS data set containing the DD names used to reference the data sets. Section 4 describes how to execute the DAS1415 module.

- You **cannot** add data set names to USOURCE(DASGUIDE) after you have executed MXG or MICS for any particular day, because these data set names will not have been processed by the DAS1415 module. Since the data set names were not processed by the DAS1415 module, the CPExpert modification to MXG or MICS would not select data set information while MXG or MICS updated your performance data base.

- You **can** delete data set names or change the response objectives for data set names after your performance data base has been updated. For example, you may wish to do this between successive runs of the DASD Component of CPExpert. The DASD Component will analyze DASD performance based upon current information in USOURCE(DASGUIDE). This is possible because the DASD Component processes USOURCE(DASGUIDE) to acquire current information. This information (data set names and response objectives) is used to guide the analysis of your performance data base.

- The VOLSER exclude or select logic described in Section 3 (Chapter 2.3 and Chapter 2.4, respectively) can be used to exclude or select specific volumes, without regard to the data set name processing.

Chapter 5: Defining Multiple PDBs

The Multiple Performance Data Base (PDB) section of the DASGUIDE module can be used to define multiple PDBs to CPExpert.

The multiple PDB definition is optional; you do not have use or define multiple PDBs. This feature is applicable only to users who wish to use the DASD Component to analyze shared DASD contention problems (see Section 2, Chapter 2.5) **and** you have multiple performance data bases that contain information relating to systems that share DASD.

Exhibit 2-11 illustrates a sample definition of multiple performance data bases.

```
*****************************************************************;
* MULTIPLE PERFORMANCE DATA BASE (PDB)  DEFINITIONS
          ;
*****************************************************************;
%LET MULTIPDB =Y;
%LET PDBLIB2 =    PDBLIBB;
%LET PDBLIB3 =    PDBLIBC;
*****************************************************************;
```

SAMPLE DISPLAY OF CPEXPERT.USOURCE(DASGUIDE) MODULE

EXHIBIT 2-11

You should specify **%LET MULTIPDB=Y;** in prefix.CPEXPERT.USOURCE(GENGUIDE) to tell CPExpert to process multiple performance data bases. You may then specify up to eight performance data bases in addition to the standard performance data base described by the PDBLIB DD statement, for a total of nine performance data bases.

You identify the DD statements that describe the additional performance data bases by specifying **%LET PDBLIBn = ddname;** , where “n” ranges from 2 through 9. In the example shown in Exhibit 2-11, the second performance data base is identified by the DD name of PDBLIBB, and the third performance data base is identified by the DD name of PDBLIBC. You can use any valid DD name to describe the performance data bases. The definitions must be in numerical order (that is, the first additional PDB must be described by PDBLIB2, the second additional PDB must be described by PDBLIB3, etc.).

